Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2014) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

23rd International Meshing Roundtable (IMR23)

Mesh Infrastructure for Coupled Multiprocess Geophysical Simulations

Rao V. Garimella[a,*], William A. Perkins[b], Mike W. Buksas[c], Markus Berndt[a], Konstantin Lipnikov[a], Ethan Coon[a], John D. Moulton[a], Scott L. Painter[a]

[a]*Los Alamos National Laboratory, Los Alamos, NM USA*
[b]*Pacific Northwest National Laboratory, Richland, WA USA*
[c]*Stellar Science Ltd. Co., Albuquerque, NM USA*

Abstract

We have developed a sophisticated mesh infrastructure capability to support large scale multiphysics simulations such as subsurface flow and reactive contaminant transport at storage sites as well as the analysis of the effects of a warming climate on the terrestrial arctic. These simulations involve a wide range of coupled processes including overland flow, subsurface flow, freezing and thawing of ice rich soil, accumulation, redistribution and melting of snow, biogeochemical processes involving plant matter and finally, microtopography evolution due to melting and degradation of ice wedges below the surface. In addition to supporting the usual topological and geometric queries about the mesh, the mesh infrastructure adds capabilities such as identifying columnar structures in the mesh, enabling deforming of the mesh subject to constraints and enabling the simultaneous use of meshes of different dimensionality for subsurface and surface processes. The generic mesh interface is capable of using three different open source mesh frameworks (MSTK, MOAB and STKmesh) under the hood allowing the developers to directly compare them and choose one that is best suited for the application's needs. We demonstrate the results of some simulations using these capabilities as well as present a comparison of the performance of the different mesh frameworks.
© 2014 The Authors. Published by Elsevier Ltd.
Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Keywords: Mesh infrastructure; coupled multiphysics simulations; terrestrial arctic; permafrost

1. Introduction

Simulations of many real-world problems involve solution of a coupled system of non-linear equations. For PDEs in the system, commonly used numerical techniques are the finite element [1,2], finite volume [3] and finite difference [4] methods. Most of these techniques use a mesh or tesselation of the domain. As is well known, the individual subdomains in the tesselation are called elements or cells while the corners of the cells are called nodes or vertices. When the topology and connectivity of the cells of a mesh is regular, the mesh is called structured; when cells can be of general topology or can be connected in arbitrary ways, the mesh is called unstructured.

*Corresponding author. Tel: +1-505-665-2929
E-mail address: rao@lanl.gov

1877-7058 © 2014 The Authors. Published by Elsevier Ltd.
Peer-review under responsibility of organizing committee of the 23rd International Meshing Roundtable (IMR23).

Structured mesh simulations do not require an extensive mesh infrastructure since the nodes and cells can be referred to using an implicit numbering. Knowing the index of a cell, the indices of its nodes and the indices of its neighboring cells can be directly inferred. Due to these implicit topological relationships between entities, the numerical discretization of PDEs on these meshes also uses a fixed predetermined template. The numbering of entities in an unstructured mesh, on the other hand, can be quite arbitrary and there are no implicit relationships between entity indices. Therefore, an unstructured mesh require explicit representation of the relationship between its entities using data structures with multiple levels of data indirection. This discussion will be focused only on unstructured mesh simulations.

Good mesh infrastructure must facilitate the import, query, manipulation and export of distributed mesh data through easy-to-use, flexible interfaces. Of these tasks, it is very important for mesh import, export and modification to be scalable in parallel because they involve the whole mesh and generally require parallel communication. On the other hand, adjacency query functions, which are localized, must be extremely efficient because they are called repeatedly at each time step during the assembly process; scalability is typically not of concern here since local adjacency queries in a well designed mesh representation do not involve parallel communication.

Choosing a suitable unstructured mesh representation or an external mesh framework library for an application [5] is difficult task requiring careful analysis. This task is even more challenging for multiphysics applications because the mesh data requirements and access patterns of the different physics kernels can be quite different. In this paper, we will describe our experiences in developing mesh infrastructure for two closely related geophysical applications with tightly coupled physics according to the criteria laid out above.

2. Amanzi and the Arctic Terrestrial Simulator

DOE's Advanced Simulation Capability for Environmental Management (ASCEM) program is an innovative effort to develop the next generation predictive simulation capability for characterizing the behavior of hazardous waste at storage sites around the DOE complex [6]. As part of the ASCEM program, a software package called *Amanzi* [7] has been developed to simulate the coupled subsurface flow and reactive transport of contaminants. Amanzi consists of structured (*Amanzi-S*) and unstructured mesh (*Amanzi-U*) simulation kernels packaged together with common functionality such as problem specification, checkpointing, restarts, visualization. Amanzi was initiated as an open source community code and is built from only open-source components. It is being used to analyze multiple DOE waste disposal sites in the demonstration and verification phase and is being released for friendly testing to DOE site analysts.

Amanzi has also been used as the basis for the development of a new software package called the *Arctic Terrestrial Simulator (ATS)* [8] for studying the effects of long term climate warming on the arctic permafrost. It is estimated that there are up to 1700 gigatons of carbon stored in the arctic permafrost and a permanent thawing due to climate change can cause a release of greenhouse gases that will dwarf existing global emissions [9]. Simulation of the arctic is a complex, coupled problem that involves overland flow, subsurface flow, freezing and thawing of ice rich soil, accumulation, redistribution and melting of snow, biogeochemical processes involving plant matter and finally, microtopography evolution due to melting and degradation of permanent ice wedges below the surface. The coupling between the physics process kernels (PKs) is managed by a heirarchy of multi-process coordinators (MPCs) and the interdependency of variables by a state manager (See Figure 2). The subsurface flow problem involves the full 3D mesh and the overland flow uses a flattened version of the top surface mesh with active feedbacks between the two. Meanwhile the freezing and thawing of soil, soil subsidence, the snow accumulation, freezing of surface water, snow melt and the biogeochemistry is computed on one dimensional vertical columns of cells. ATS maintains a sophisticated directed acyclic graph of dependencies between processes and variables in order to dynamically manage process evolution in different scenarios. Such an involved simulation requires a capable mesh infrastructure that can serve up mesh data in different ways based on the needs of the physics process kernels.

Process kernels in Amanzi and ATS primarily use the Mimetic Finite Difference (MFD) method [11,12] or the finite volume method [3] with two-point flux approximation to discretize the governing equations except in processes in which the lateral variation can be ignored and the equations solved in each vertical column of cells independently. Amanzi supports boundary conditions, initial conditions and material properties based on entity sets that are labeled

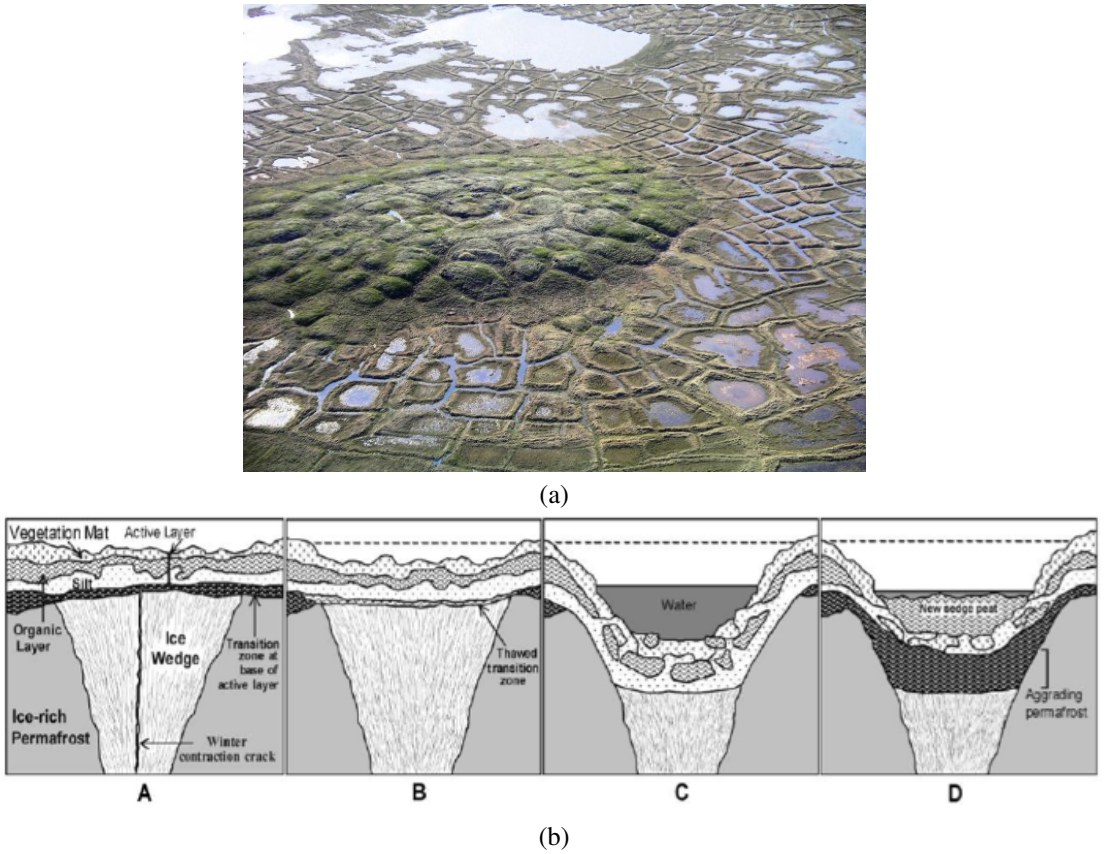

Fig. 1. (a) Photograph of a melting pingo and polygon wedge ice near Tuktoyaktuk, Northwest Territories, Canada (Courtesy Wikipedia: `http://en.wikipedia.org/wiki/Pingo`) (b) Schematic showing the degradation of ice wedges and subsequent subsidence of soil in the arctic permafrost. Adapted from Jorgenson [10]

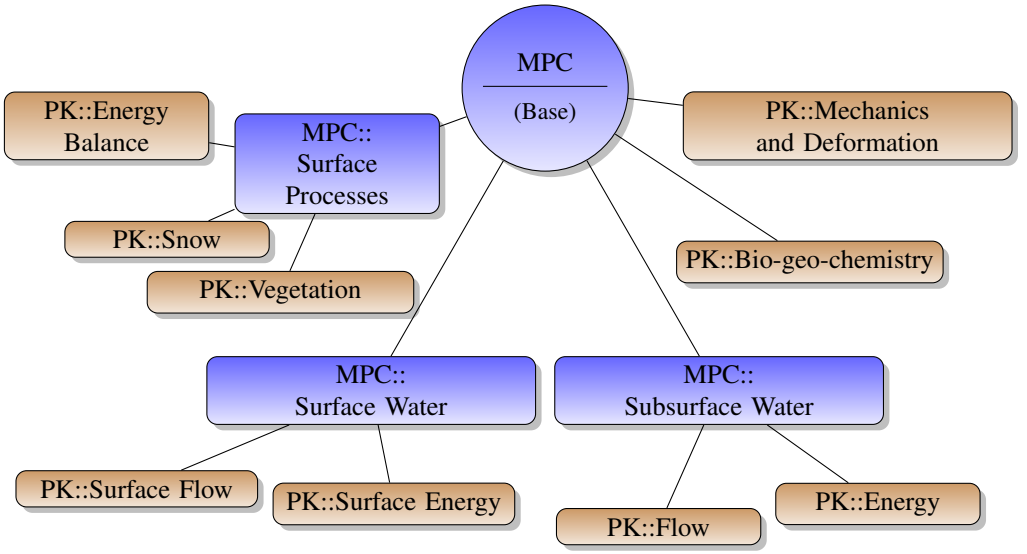

Fig. 2. Hierarchical multi-process coordinators (MPCs) coupling the process kernels (PKs) executing the individual physics in Amanzi-ATS

a priori or constructed within the simulation using geometric constructs. It also includes capabilities for pointwise extraction of observations, checkpointing, restarts and visualization.

Amanzi is built using over 20 third party libraries including packages for matrix solution, mesh partitioning, mesh management, geochemistry, parallel communication, XML parsing and unit testing. Amanzi and all its third party libraries can be built automatically using a bootstrap script that invokes CMake [13] or autotools [14] as necessary. Amanzi and ATS have been tested on a wide range of platforms ranging from laptops to supercomputers with thousands of cores.

In the rest of this discussion, we will refer to the combined software product consisting of ATS physics kernels and Amanzi's shared capabilities, including the mesh infrastructure, as *Amanzi-ATS*.

3. Mesh infrastructure in Amanzi-ATS

The Amanzi-ATS mesh infrastructure handles general polygonal and polyhedral meshes in support of the MFD method. At the same time, the Amanzi-ATS mesh infrastructure can be queried if a mesh is a regular hexahedral mesh so that appropriate simplifications can be employed in the numerical methods for a faster solution.

The mesh infrastructure supports three types of mesh entities: *cells* which are the highest dimensional entities in the mesh, *nodes* which are the lowest dimensional entities and *faces* which are intermediate entities that are actively used in the MFD method ("faces" for a surface mesh are topologically one-dimensional entities or edges). The Amanzi-ATS mesh infrastructure supports a variety of upward and downward adjacency queries such as the faces of a cell (including their directions with respect to the cell center), nodes of a cell and cells connected to a face. Other less used queries are cells connected to a node, face-connected neighbors of a cell and node-connected neighbors of a cell. Given a cell, it is also possible to query if the cell is a standard element type like a triangular prism or hexahedron, or a general polyhedron. It is possible to query the spatial dimension of the mesh and the topological dimension of cells in order to construct appropriately sized local matrices.

Another important feature of the Amanzi-ATS mesh infrastructure is that entities are referred to solely by their local numeric identifier or *ID* and not through object handles that encode their ID, type and other miscellaneous information. This design allows Amanzi-ATS to be far more lightweight than if one used objects. The disadvantage is that the developers and the code they write must be aware of what type of entities they are handling at any given point, which is usally not a problem. Additionally, mixed lists containing entities of different types must be augmented by additional information.

Meshes in Amanzi-ATS are designed to be distributed across multiple processors with an extra layer of cells around each partition (See Figure 3). This extra layer called the *ghost* layer and entities in the ghost layer are a copy of a master or *owned* entity on another processor. Ghost entities are typically read only - a ghost entity or its data is typically not modified except through their masters. Process kernels may request the mesh infrastructure to return *owned* entities, *ghost* entities or all entities in any adjacency query.

Most parallel communication of mesh data in Amanzi-ATS is done during mesh import and setup or during any deformation of the mesh during the simulation. Storage and parallel communication of solution variables is handled by a state manager in Amanzi-ATS, not by the mesh infrastructure. Amanzi-ATS stores solution variables in special data containers called *Composite vectors* based on *Epetra vectors* from the Trilinos software suite[15]. Epetra vectors are constructed from the distributed data to be represented and an *Epetra map*, containing the global IDs of owned entities followed by ghost entities of the partition. During a parallel communication step, the Epetra map indicates how the ghost values in Epetra vectors should be updated.

Geometric queries supported by the Amanzi-ATS mesh infrastructure include node coordinates, face normals with respect to a connected cell, face centroids and cell centroids, face areas and cell volumes. To avoid repeated recomputation, geometric quantities are cached in the mesh infrastructure layer and updated only if the mesh is modified.

The Amanzi-ATS mesh infrastructure supports querying of named sets of entities. Sets are defined as entities contained in a named region of a "geometric model" of the domain. The model is not a true geometric model since building one for geological domains is more complex than for engineering problems. Instead, it is a rudimentary container for general region definitions such as point regions, plane regions and box regions. A special region called "labeled set" region is also used indicate a predefined set of entities specified in the input mesh file and is useful to tag sets of entities on irregular surfaces prior to the simulation. The "labeled set" region is used to import predefined

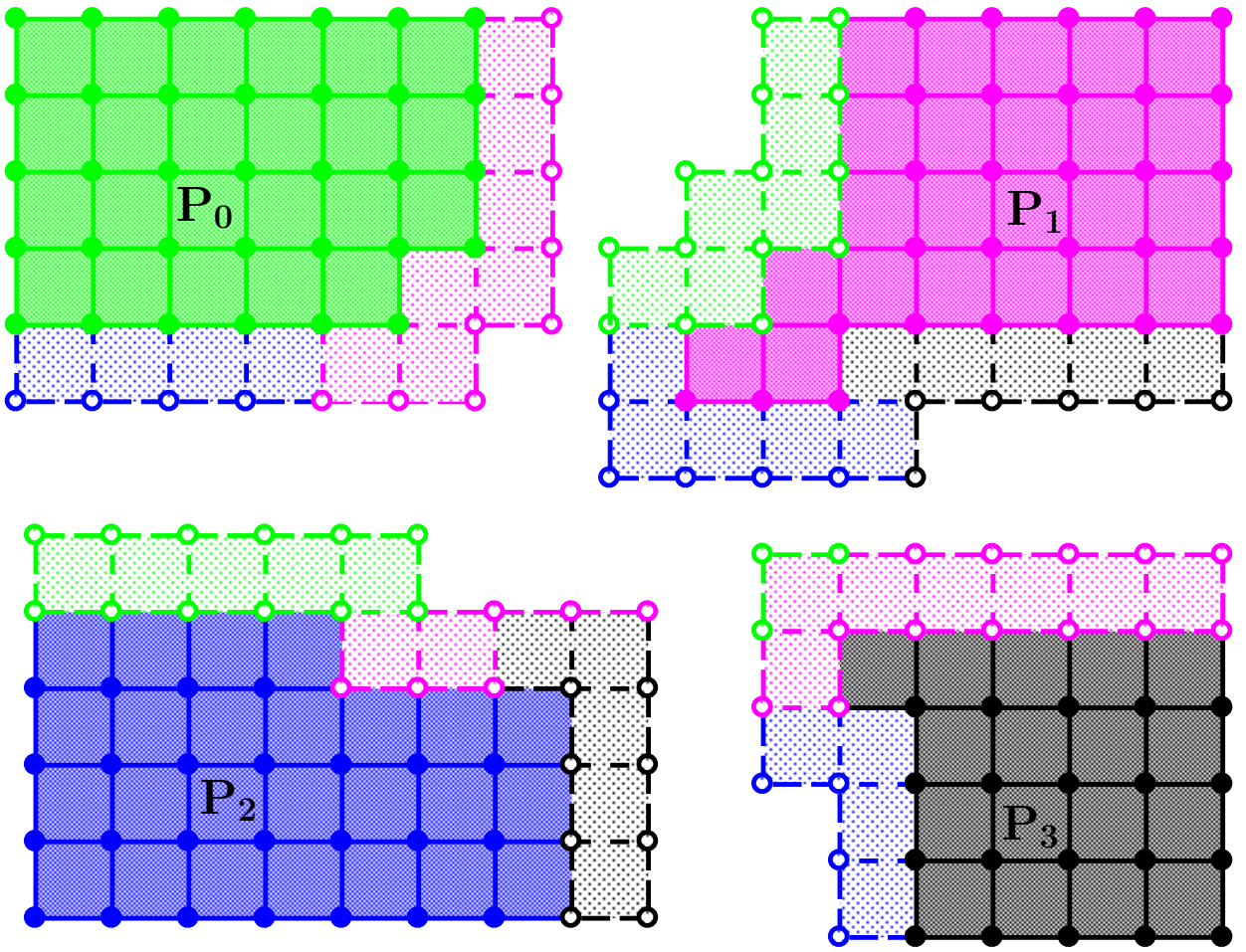

Fig. 3. A four partition parallel mesh with ghost layers used in Amanzi-ATS mesh infrastructure. Cells with dark shading are masters, with light shading are ghosts. Solid edges/faces are masters, dotted are ghosts. Solid circles are masters, dotted are ghosts. The coloring of entities indicates which processor the master resides on. Note that in this schematic the lowest numbered processor sharing an entity is the master but this is not a requirement. Also, note that the cells in the ghost layer are complete and can be queried for the full set of bounding entities just like an owned cell.

sidesets (facesets) from input files. Region definitions may be combined using logical operations or expanded using sweeping and rotating. It is possible to request entities of any type on a region (except labeled sets which have a predefined type of entity) and optionally, restrict the returned set to only owned or ghost entities.

Meshes in Amanzi-ATS are typically imported using a file in the Exodus II/Nemesis I format. Exodus II [16] is a file format developed by Sandia National Laboratories to describe finite element meshes and associated data such as cell sets, face sets and element sets. Nemesis I [17] files are meant to represent distributed meshes and there is one Nemesis I file per partition. Nemesis I files are augmented Exodus II files and typically contain some extra information such as global IDs of entities Amanzi-ATS can read an Exodus II mesh on processor 0, partition it using Metis [18] or Zoltan [19] and distribute it to the different processors. Alternately, it can read Nemesis I files and use the global IDs of entities to create ghost layers and establish communication maps between processors. The latter option is more scalable than the first but eventually, an entirely different approach may have to be developed when simulations are run on tens to hundreds of thousands of partitions. Since some Amanzi-ATS process kernels solve a simplified system on columns of cells, the partitioners used for Amanzi-ATS are forced to partition meshes only along the horizontal directions; otherwise, columns may end up distributed across multiple processors.

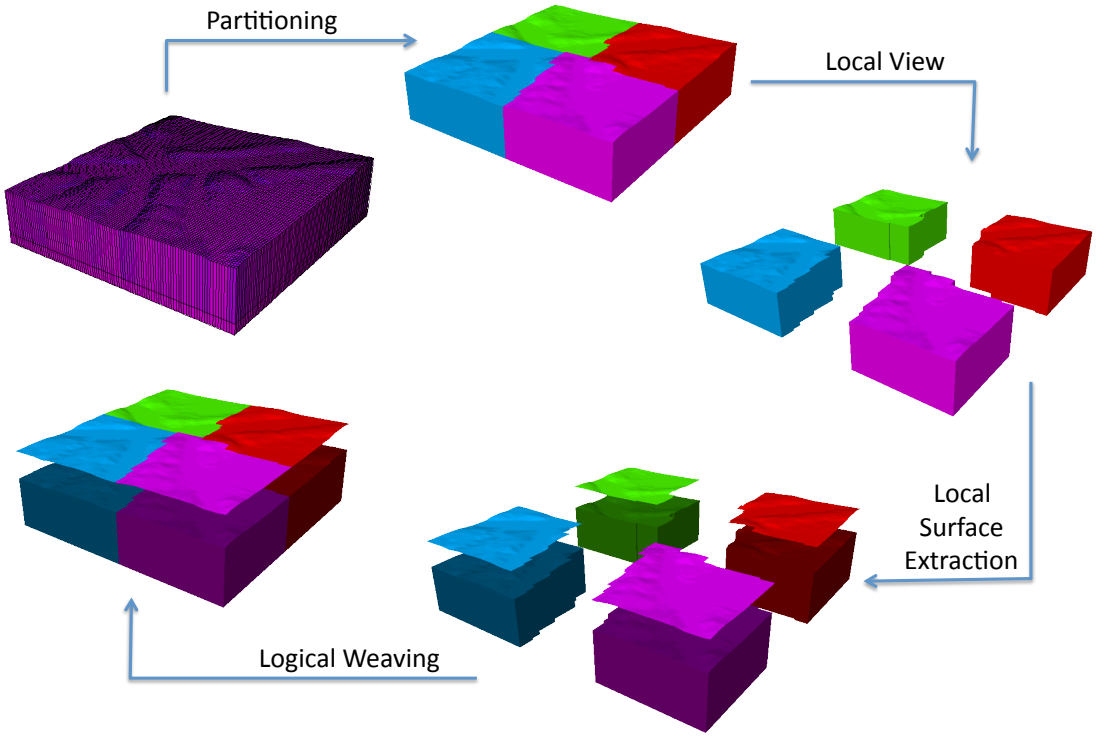

Fig. 4. Extraction of surface meshes from distributed volume meshes and establishment of parallel connections between the distributed surface meshes

Meshes in Amanzi-ATS may also be initialized through an internal mesh generator although this option is restricted to regular meshes in a rectangular domain only.

Lastly, Amanzi-ATS can create a mesh by extracting entity sets from another mesh. This option is typically used to extract surface mesh from face sets in a volume mesh although it can be used to extract a smaller volume mesh using a cell set. If requested, extracted surface meshes can be flattened from a manifold to a planar two-dimensional mesh. When extraction of submeshes is done on a distributed mesh, the extracted partitions are "woven" back together in a post-processing step to provide the appropriate parallel connectivity (See Figure 4). Extracted meshes maintain a memory of their parent entity enabling coupled simulations on the parent and extracted meshes.

After the import process, the mesh infrastructure in Amanzi-ATS can collapse degenerate edges and any attached degenerate elements. During this process some elements that may have been standard elements like hexahedra or triangular prisms become general polyhedra. The ability to collapse degenerate edges is a very useful feature for handling geological meshes where layers may become pinched out because surfaces come too close together. This process also updates all mesh entity sets by removing any entities that have been removed from the mesh.

In addition, process kernels may request the mesh infrastructure to build columns of elements (if possible), and return information about the cell above and cell below. This is useful for process kernels in which the vertical behavior of the system is predominant and is largely decoupled in the horizontal direction. This simplification greatly increases the efficiency of several process kernels such as snow compaction, soil subsidence and biological degradation of organic matter.

A recent capability in Amanzi-ATS mesh infrastructure is the ability to deform meshes as part of the soil subsidence due the melting of ice wedges. This causes the volume of some ice wedge cells to reduce and the cells of soil above them to shift down. The process kernel provides the mesh infrastructure with target volumes for a small subset of cells and the minimum volumes for all cells of the mesh (based on the soil properties). The deformation algorithm uses a minimization approach to move the nodes of affected cells and their neighbors in order to reach the target volumes. The

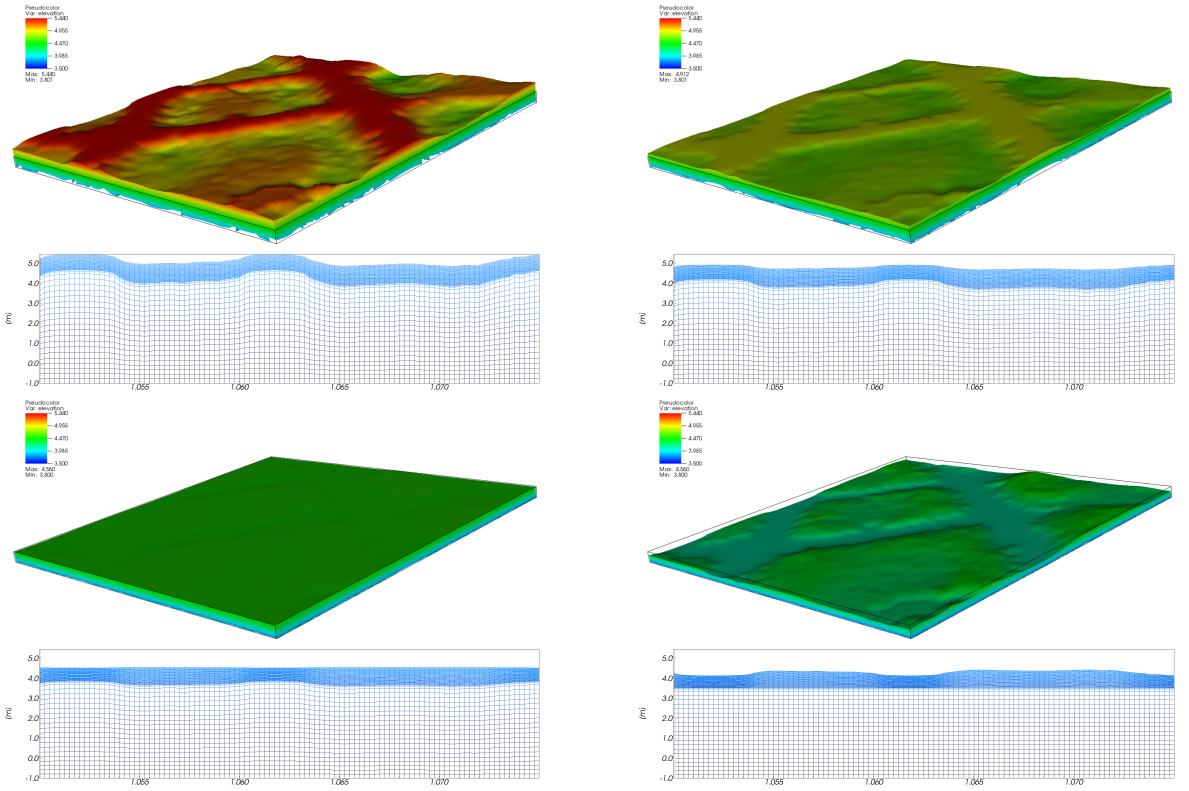

Fig. 5. Example of deformation of a low-centered polygonal ground to a high-centered polygonal ground. *Note that this example was not driven by ice-wedge melting and soil subsidence in a simulation but by prescribed displacements instead.* Image courtesy of G. Manzini, LANL

movement of nodes is constrained to be in the downward direction only to prevent uplift of any neighboring columns of cells. Changes in node positions are communicated across processors to ghost nodes. Geometric quantities like cell volumes are recomputed after deformation. This functionality is undergoing standalone testing while coupled ice melt and soil subsidence models are being developed to drive the deformation (See Figure 5).

4. External mesh frameworks

Amanzi-ATS's mesh infrastructure was designed to be a lightweight layer over an existing mesh framework library rather than a reinvention of functionality available elsewhere. Most of the detailed mesh description is stored in the external mesh framework and Amanzi-ATS's mesh infrastructure merely recasts the data into a suitable form with minimal overhead. This design principle also drives the use of entity IDs to communicate with the Amanzi-ATS process kernels rather than pointers to entity objects, as mentioned in the last section.

A large number of mesh frameworks were surveyed and considered for use in Amanzi-ATS (e.g, LibMesh [20], GMDS [21], VTK [22], ITAPS/iMesh [23], GRUMPP [24], NWGrid [25], OpenMesh [26], FMDB [27]) but were excluded either because they did not support distributed meshes, did not support polyhedral elements or because they were not appropriately open-sourced. As a result, Amanzi-ATS currently uses MSTK [28] from LANL, STK-Mesh [29] from SNL or MOAB [30] from ANL as the underlying frameworks. During a simulation only one of these frameworks is active but the framework to be used can be selected at runtime.

MSTK is an unstructured mesh framework for general meshes that uses a flexible mesh representation (Amanzi-ATS uses only the full heirarchical representation of MSTK dubbed as the F1 representation [5]). MSTK can represent multiple meshes of different dimensionality simultaneously and represents standard elements as well as general polygonal and polyhedral elements. MSTK can answer queries for any upward or downward adjacency in the mesh without the need a global search. MSTK handles distributed meshes supporting a complete layer of ghost elements around

partitions. It offers support for mesh attributes for field data as well as named sets of mesh entities. Mesh modification is available in the form of node repositioning as well as edge collapses, edge splits and a few other functions. Currently, mesh modification operations have limited parallel support.

STK-Mesh is the unstructured mesh infrastructure component of the Sierra ToolKit project at Sandia National Labs and is released as part of the Trilinos suite of software tools. It supports parallel, heterogeneous, dynamically modifiable unstructured meshes. STK-Mesh supports all standard elements types; it is unclear if support for polyhedral meshes is complete. STK-Mesh does not represent edges and faces explicitly by default but these can be created dynamically. Mesh elements are organized into blocks and furthermore, buckets of a single element type. STK-Mesh allows for storage of field data associated with mesh entities and tightly couples the organization of the two. Since buckets contain elements of a single mesh type, processing of field data on a bucket is designed to be more vectorizable. Like MSTK, STK-Mesh supports distributed meshes with one layer of ghost entities. STK-Mesh supports modification of distributed meshes. One disadvantage with STK-Mesh is that there is no default mesh class and the application has to build a custom mesh class using mesh meta data, mesh bulk data and field data.

MOAB is a mesh infrastructure library capable of representing structured and unstructured meshes consisting of standard finite elements as well as polygons and polyhedra. MOAB is primarily geared towards mesh query rather than mesh modifications. A unique feature of MOAB is that allows for queries on a range of mesh entities for greater efficiency instead of only one entity at a time. The default MOAB representation contains cells and nodes; faces and edges are created upon request. MOAB supports distributed meshes with options to create a layer of ghost elements around each partition. It also supports storing of field data on mesh entities using tags as well as the creation of entity sets. MOAB implements the ITAPS iMesh interface which is a standardized mesh interface implemented by several packages as part of the SciDac TSTT (`http://www.scidac.gov/ASCR/ASCR_TSTT.html`) effort. MOAB cannot read Nemesis I files like MSTK and STK-Mesh. Instead, a a MOAB specific utility must be used to partition a serial mesh resulting in a single MOAB specific HDF5 file with the partition information. This file can be read by MOAB when importing the mesh in parallel.

The three frameworks support varying degrees of the Amanzi-ATS mesh interface, with MSTK supporting all the functionality followed by STKmesh and then MOAB. Most notably, two dimensional mesh support, mesh extraction, elimination of degenerate elements and mesh deformation are not implemented with STK-Mesh and MOAB. A mesh factory class in Amanzi-ATS allows for dynamic selection of the particular framework to use for a given simulation.

5. Performance profiling and comparisons

The use of three alternative mesh framework libaries under a single mesh interface in Amanzi-ATS offers the unique opportunity to compare the performance of the three mesh framework libaries directly and gain some unique insights into their capabilities. To the best of our knowledge, this is the first time a direct comparison of the perfomance of three mesh infrastructure libraries has been published.

Since the STK-Mesh and MOAB implementations of the Amanzi-ATS mesh infrastructure did not support extraction of meshes, it was not possible to compare the performance of a coupled simulation of the terrestrial arctic. Instead, a subsurface flow (governed by Richards equation [31]) and contaminant transport problem was chosen for the comparison. No geochemistry was involved in the problem. Although this is essentially a two-dimensional problem in x- and z-directions, it was simulated in a three-dimensional domain of unit thickness in the y-direction. Accordingly, the domain extended from the origin to (216.0,1.0,107.52) meters and a geometrically "structured" mesh with 432, 1 and 256 elements along the x-, y- and z-axes was used. The mesh was distributed across 8 processors for the simulation. The domain had three geological layers with different material properties. The full problem involves ramping up the flow to steady state over nearly 2000 years and then turning on and off contaminant release to study the problem hundreds of years into the future. However, recognizing that the interaction of the analysis code with the mesh infrastructure does not change much over the span of the simulation, only 100 years of the computation ramping up to steady state (involving about 200 time steps) were conducted. MSTK version 2.11rc5, STK-Mesh from Trilinos version 11.6.1 and MOAB version 4.6.0 were used in the comparison.

The code was compiled using optimization with source information (-O3 -g) and profiled using HPC Toolkit suite of tools from Rice University [32]. Three runs were conducted for each framework in the study in order to avoid any anomalous results and all runs were performed on a single machine under no other load. Performance counters

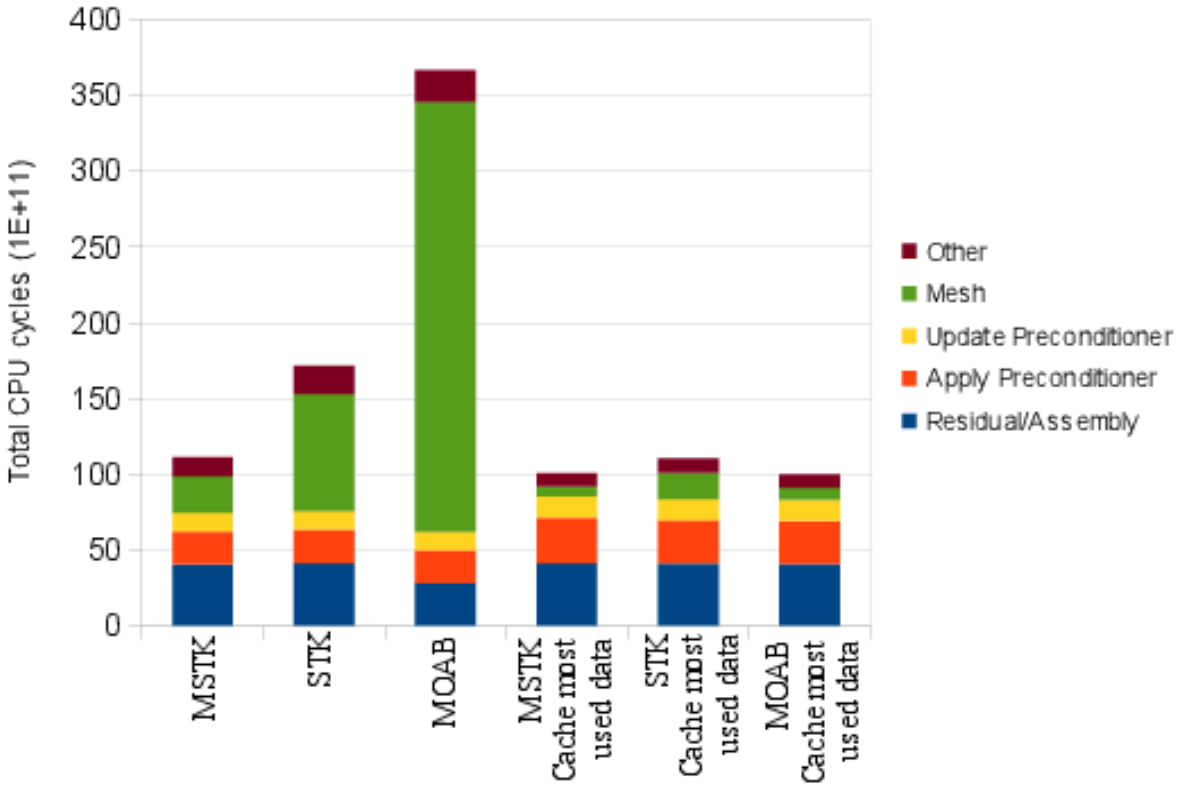

Fig. 6. Performance of MSTK, STK-Mesh and MOAB without and with caching of most frequently used topological data separated by major components of the simulation of a subsurface flow and transport simulation using Amanzi.

to measure the total CPU cycles taken by various parts of the code and the cycles lost in L2 cache misses were used. The former gives a good idea of the computational cost of the various sections of the code whereas the latter indicates whether data in the mesh framework is laid out efficiently thereby avoiding cache misses and minimizing slow accesses from main memory.

The mesh operators are primarily invoked during initialization and during matrix assembly (for preconditioner update and residual calculation). With a sufficiently long simulation, the initialization costs become insignificant compared to the repeated cost of mesh adjacency retrieval. The results of the performance analysis were carefully parsed to separate out the cost of the mesh from the assembly and solver sections. The costs of the different parts of simulation for the three frameworks are shown in Figure 6 (first three columns) for the case when all topological data in the mesh is directly retrieved from the underlying mesh framework.

From the results it clear that MSTK has outperformed STK-Mesh and MOAB significantly. A detailed analysis of the performance cost of the mesh infrastructure (not shown) revealed that most of the cost is in accessing faces and face directions of cells during matrix assembly in the MFD method, and cells of a face during upwinding [33]. The effect is more pronounced for STK-Mesh and MOAB mesh which do not support faces natively; in addition, the MOAB mesh framework in the version used was confirmed by the developers to be slow in accessing local adjacency information (but is being fixed in an upcoming version).

Even with the superior performance of MSTK in mesh adjacency accesses, mesh queries can be seen to be a significant portion of the total cycle (10-15%). A careful analysis of the results showed that most of the cost of mesh queries came from L2 cache misses rather than computation. The results for STK-Mesh were similar while MOAB had somewhat fewer misses. The reason for the high number of cache misses, particularly in MSTK, is several levels of indirection necessary to access data needed by the highest level caller of adjacency queries. For example, the

MSTK	STK-Mesh	MOAB
165 MB	195 MB	240 MB

Table 1. Average memory usage after initial setup as stated by Valgrind tool Massif for the three mesh frameworks used in Amanzi-ATS

Amanzi-ATS infrastructure requests local IDs of faces bounding a given cell ID. The MSTK version of the code for this operator first gets a pointer to the cell object from the cell ID. It then retrieves a list of pointers to face objects stored in the cell object. Finally, it peers into the face objects to retrieve their IDs and puts the IDs into a list that is returned to the calling routine. These multiple levels of indirection prove to be very cache inefficient since the data required may be scattered over a wide swath of memory and not necessarily organized in a linear fashion. This example holds a lesson for the future design of internal data structures of mesh frameworks and that is to organize data in linear data structures to the extent possible while still hiding the data from applications using functional interfaces. Doing so holds particular challenges for applications that peform topological modification of the mesh but it is unavoidable as newer computer architectures require applications to use memory and computational capacity more efficiently.

In order to mitigate the cost of the most frequently accessed adjacencies, Amanzi-ATS has the option of caching some adjacency information at the generic mesh infrastructure level. The performance results with the three frameworks in the presence of adjacency caching is shown in the last three columns in Figure 6. Clearly, the overhead of the mesh infrastructure is greatly reduced for all three frameworks and differences in the framework performance become inconsequential. While such a duplication of data is to be avoided in general, it is a reasonable strategy at this life stage of the Amanzi-ATS code as the numerical solution strategies are somewhat stable. Still, this is not a blanket recommendation for any multi-physics code and frequent re-evaluation of code performance is warranted to see which strategies should be used.

In addition to performance data, the memory usage of Amanzi-ATS runs with the three frameworks was measured using Massif in the Valgrind suite of tools (`http://www.valgrind.org` [34]). Since this is a very time consuming process only one run was performed with each framework with no-caching enabled. In addition, one run using MSTK and caching of data was performed to measure the increase in memory usage due to the extra data. Both heap and stack memory was measured. The initial memory usage during setup was ignored; rather average memory usage for times when the simulation is merely advancing in time is considered. The results are shown in Table 1 and in this too, MSTK appears to be advantageous over the other two frameworks. The caching of variables in the Amanzi-ATS mesh infrastructure resulted in approximately 15 MB increase in memory usage.

It should be noted that even though MSTK had a slight advantage, having been developed by the primary author, a reasonable effort was made to use the other two frameworks efficiently as well. Finally, private communications with the developers of STK-Mesh and MOAB have revealed that the issues raised in this comparison are currently being worked on actively and are expected to be mitigated in upcoming releases of both libraries.

6. Discussion

This paper presented the design and implementation details of the mesh infrastructure developed for a multiphysics simulation code in the area of geophysics. Three different mesh frameworks, MSTK, STK-Mesh and MOAB were used under a generic mesh interface invoked by the rest of the code. This permitted a direct comparision of the computational and memory costs of the three frameworks. MSTK was seen to have an advantage over the two other frameworks in computational cost and memory usage.

One interesting demonstration of this study were that complex indirection of data causes performance killing cache misses. This should lead to better design of internal data structures of the frameworks. Another demonstration is that a rich topological representation of the mesh is not condemned to be bloated and should be used whenever possible.

Future work involves conducting scaling studies on Amanzi-ATS, verifying and testing the mesh deformation as part of a coupled simulation and further reducing the mesh access cost.

Acknowledgements

This work was performed under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396 and supported by the Lab Directed Research and Development program. LA-UR-14-24112.

References

[1] T. Hughes, The Finite Element Method: Linear Static and Dynamic Finite ELement Analysis, Prentice Hall, 1987.
[2] J. Reddy, An Introduction to the Finite Element Mehod, McGraw-Hill, 2005.
[3] R. Eymard, R. Herbin, Handbook of Numerical Analysis, Vol VII, North-Holland, 2000, pp. 713–1020.
[4] O. Rübenkönig, The Finite Difference Method (FDM) - An introduction, Albert Ludwigs University of Freiburg, 2006.
[5] R. Garimella, Mesh data structure selection for mesh generation and FEA applications, International Journal of Numerical Methods in Engineering 55 (2002) 451–478.
[6] M. Williamson, J. Meza, D. Moulton, I. Gorton, M. Freshley, P. Dixon, R. Seitz, C. Steefel, S. Finsterle, S. Hubbard, M. Zhu, K. Gerdes, R. Patterson, Y. Collazo, Advanced Simulation Capability for Environmental Management (ASCEM): An overview of initial results, Technology and Innovation 13 (2011) 175–199. DOI: 10.3727/1949822411Z13085939956625, http://ascemdoe.org.
[7] P. R. Dixon, J. D. Moulton, I. Gorton, J. Meza, M. Freshley, Amanzi and Akuna: Two new community codes for subsurface contaminant flow and transport, in: American Geophysical Union, Fall Meeting Abstracts, 2011, p. L5. Http://software.lanl.gov/ascem/trac/.
[8] E. Coon, M. Berndt, R. Garimella, J. D. Moulton, G. Manzini, S. L. Painter, Computational Advances in the Arctic Terrestrial Simulator: Modeling Permafrost Degradation in a Warming Arctic, AGU Fall Meeting Abstracts (2013) A2195.
[9] C. Tarnocai, J. G. Canadell, E. A. G. Schuur, P. Kuhry, G. Mazhitova, S. Zimov, Soil organic carbon pools in the northern circumpolar permafrost region, Global Biogeochemical Cycles 23 (2009).
[10] M. T. Jorgenson, Y. L. Shur, E. R. Pullman, Abrupt increase in permafrost degradataion in arctic Alaska, Geophysical Research Letters 33 (2006).
[11] J. M. Hyman, M. Shashkov, Adjoint operators for the natural discretizations of the divergence, gradient and curl on logically rectangular grids, Applied Numerical Mathematics 25 (1997) 413 – 442.
[12] K. Lipnikov, G. Manzini, M. Shashkov, Mimetic finite difference method, Journal of Computational Physics 257, Part B (2014) 1163 – 1227. Physics-compatible numerical methods.
[13] K. Martin, B. Hoffman, Mastering CMake, 6 ed., Kitware, Inc., 2013. URL: `http://cmake.org`.
[14] J. Calcote, Autotools: A Practioner's Guide to Autoconf, Automake and Libtool, No Starch Press, 2010. URL: `https://www.gnu.org/software/automake/`.
[15] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, K. Stanley, An overview of the trilinos project, ACM Trans. Math. Softw. 31 (2005) 397–423.
[16] L. Schoof, V. Yarberry, Exodus II: A finite element data model, Technical Report SAND92-2137, Sandia National Laboratories, 1996.
[17] G. L. Hennigan, J. N. Shadid, Nemesis I: A set of functions for describing unstructured finite-element data on parallel computers, Technical Report, Sandia National Laboratories, 1998.
[18] G. Karypis, V. Kumar, MeTis: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version 4.0, Online at `http://www.cs.umn.edu/ metis`, 2009.
[19] E. G. Boman, U. V. Catalyurek, C. Chevalier, K. D. Devine, The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, Ordering, and Coloring, Scientific Programming 20 (2012).
[20] B. S. Kirk, J. W. Peterson, R. H. Stogner, G. F. Carey, `libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations, Engineering with Computers 22 (2006) 237–254.
[21] F. Ledoux, J. C. Weill, Y. Bertrand, GMDS: A generic mesh data structure, in: 17th International Meshing Roundtable, 2008. Research Note in `http://www.imr.sandia.gov/papers/imr17/Ledoux.pdf`.
[22] W. Schroeder, K. Martin, B. Lorensen, The Visualization Toolkit, 4 ed., Kitware, Inc., 2006. ISBN 978-1-930934-19-1 `https://www.vtk.org`.
[23] C. Olivier-Gooch, L. Diachin, M. S. Shephard, T. Tautges, J. Kraftcheck, V. Leung, X. Luo, M. Miller, An interoperable, data-structure-neutral component for mesh query and manipulation, ACM Transactions on Mathematical Software 37 (2010).
[24] C. Olivier-Gooch, GRUMMP User's Guide, Technical Report, Department of Mechanical Engineering, The University of British Columbia, 2010.
[25] H. E. Trease, L. L. Trease, B. Riley, NWGrid Users Manual, Technical Report, Pacific Northwest National Laboratory, 2000. `http://www.emsl.pnl.gov/nwgrid`.
[26] L. Kobbelt, OpenMesh 3.1 Documentation, Technical Report, RWTH Aachen University, 2014. `http://www.openmesh.org`.
[27] E. Seol, M. Shephard, Efficient distributed mesh data structure for parallel automated adaptive analysis, Engineering with Computers 22 (2006) 197–213.
[28] R. Garimella, MSTK - a flexible infrastructure library for developing mesh based applications, in: Proceedings of the 13th International Meshing Roundtable, 2004, pp. 213–220.
[29] H. Edwards, T. Coffey, D. Sunderland, A. Williams, STK-Mesh tutorial minisymposium, Online, 2011. SAND2011-0814C `http://trilinos.sandia.gov/packages/stk/STK_Tutorial_1.pdf`.

[30] T. J. Tautges, R. Meyers, K. Merkley, C. Stimpson, C. Ernst, MOAB: A Mesh-Oriented Database, SAND2004-1592, Sandia National Laboratories, 2004. Report.
[31] L. Richards, Capillary conduction of liquids through porous mediums, Physics 1 (1931).
[32] L. Adhianto, S. Bannerjee, M. Fagan, M. Krentel, G. Marin, J. Mellor-Crummey, N. R. Tallent, HPCToolkit: Tools for performance analysis of optimized parallel programs, Concurrency and Computation: Practice and Experience 22 (2010) 685–701.
[33] R. Courant, E. Isaacson, M. Rees, On the solution of nonlinear hyperbolic differential equations by finite differences, Communications on Pure and Applied Mathematics (1952) 243–255.
[34] N. Nethercote, R. Walsh, J. Fitzhardinge, Building workload characterization tools with Valgrind, Online at `http://valgrind.org/docs/iiswc2006.pdf`, 2006. Invited tutorial, IEEE International Symposium on Workload Characterization.